AFRL-IF-RS-TR-2006-179
Final Technical Report
May 2006

FUSELET DEVELOPMENT ENVIRONMENT (FDE)

GE Global Research

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** MAY 2006	**3. REPORT TYPE AND DATES COVERED** Final Aug 03 – Dec 05

4. TITLE AND SUBTITLE FUSELET DEVELOPMENT ENVIRONMENT (FDE)	**5. FUNDING NUMBERS** C - F30602-03-C-0193 PE - 62702F PR - IMCR TA - BA WU - 01
6. AUTHOR(S) Bart Ingleston	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) GE Global Research One Research Circle Niskayuna New York 12309	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFSE 525 Brooks Road Rome New York 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2006-179

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: James R. Milligan/IFSA James.Milligan@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA #06-357	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 Words)*

This report describes an overview of a research and development project to produce a Fuselet Development Environment (FDE) that facilitates user friendly authoring of information transformation programs called “fuselets.” project. The FDE also provides fuselet runtime support in conjunction with a Joint Battlespace Infospshere (JBI) information management system.

14. SUBJECT TERMS Joint Battlespace Infospshere (JBI), Fuselet, Interactive Development Environment, Information Transformation	**15. NUMBER OF PAGES** 15
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

1.1 Acknowledgements

The Fuselet Development Environment project was a joint effort among GE Global Research (GE GR) along with the ISX Corporation (ISX) for the Air Force Research Laboratory (AFRL) Joint Battlespace Infospshere (JBI) Fuselet Production Environment (FPE) and the Fuselet research program.

<u>GE Global Research Team</u>
Dr. Louis J. Hoebel, GE Principal
Investigator, Manager, Enterprise
Architecture Lab
Bart Ingleston, Technical Lead
Dr. Andrew Crapo
Marc Garbiras
John Lizzi
Bowden Wise
Steve Markham
Steve Linthicum
Anne Gilman
Matt Abrams

<u>ISX Team</u>
Justin Donnelly, ISX Principal
Investigator
Brady Tsurutani
Matt Greenberg

1.2 Contact Information

Louis J. Hoebel, GE Principal Investigator
GE Global Research
1 Research Circle
Niskayuna, NY 12309
Phone: 518-387-6870; Fax: 518-387-6104; E-Mail: hoebel@research.ge.com

Justin Donnelly, ISX Principal Investigator
ISX Corporation
Camarillo Business Center
760 Paseo Camarillo, Suite 401
Camarillo, CA 93010
Phone: 805-484-6132; Fax: 484-6195; E-Mail: jdonnelly@isx.com

James R. Milligan
Systems & Information Interoperability
AFRL/IFSE
525 Brooks Road
Rome, NY 13441-4505Phone: 315-330-1491; Fax: 315-330-8281; E-Mail: James.Milligan@rl.af.mil

1.3 Report Overview

This is the final report for the Fuselet[1] Development Environment (FDE) project. It provides an overview the FDE project, details some of the lessons learned, and proposes future directions for the FDE with respect to the AFRL JBI and JBI Fuselet research programs.

2 Project Summary

The objective of this effort was to develop a fuselet production environment for the Joint Battlespace Infosphere (JBI). The scope of this effort was to research, develop, document, integrate, install, demonstrate, and deliver a fuselet production environment that consists of a Fuselet Development Environment (FDE), a Fuselet Library containing a collection of predefined fuselet components, a Fuselet Scripting Language (FSL), and a fuselet execution and testing Sandbox.

2.1 FDE Project Overview

The GE Global Research and ISX Corporation team[2] has developed a fully functional Fuselet Development Environment (FDE) that allows users to easily and rapidly build, test and deploy JBI fuselets. The FDE provides tools and features to create solutions to recurring information tasks and enables JBI process integration. The GE/ISX team focused on delivering a graphical authoring environment for the production of the most

[1] For more information about fuselet technology, visit http://www.fuselet.org
[2] Referred to in this documents at the GE/ISX team

commonly required, and highest value, fuselets and fuselet components. The GE/ISX team also delivered a fuselet execution and test platform and provided a connection to any JBI Common Application Programming Interface (Common API, or CAPI)-compliant platforms made available by AFRL or others. The FDE provides solutions to recurring information tasks and enables JBI process integration. The FDE is built on and is delivered as plug-ins to the commercially supported, open source Eclipse Platform[3].

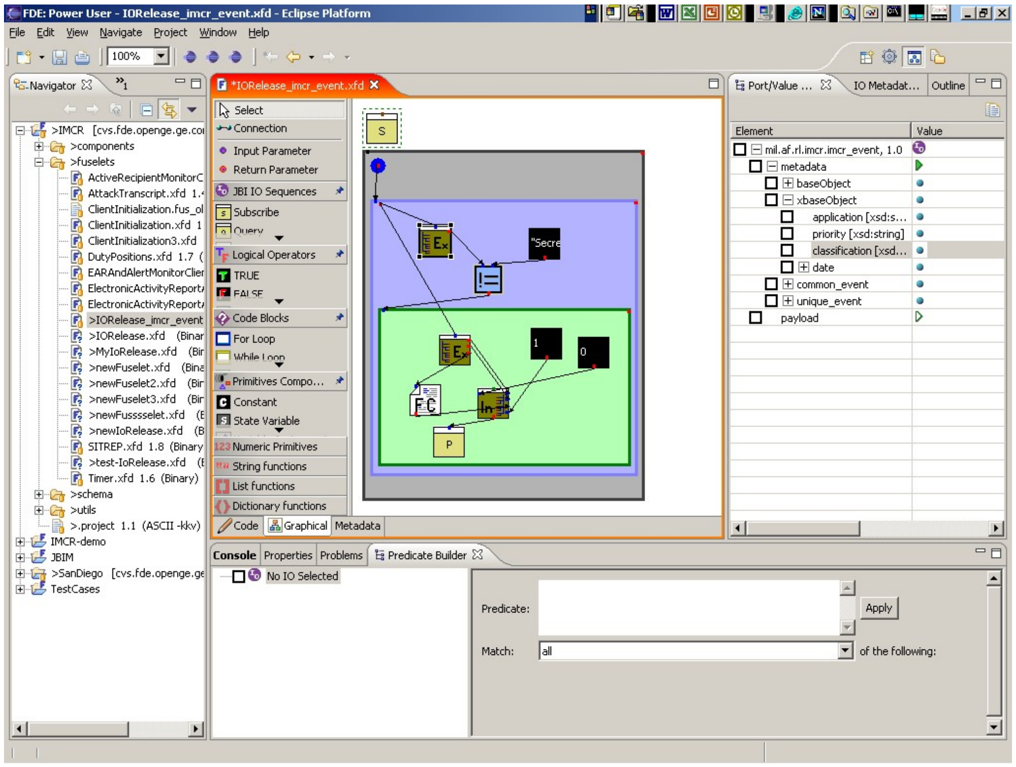


Figure 1: FDE Workbench (development mode)

2.2 FDE Capabilities

The FDE capabilities can be split into two general functional categories; *development* and *runtime*. The following capabilities are presented under the mode in which they are primarily used. See the ***Fuselet Development Environment User's Guide*** for a detailed description of all the of the FDE capabilities to include the Fuselet Scripting Language, Graphical Development Editor, Fuselet Execution Harness, Simulation Sequencer, and Fuselet Debugger. Each of these components is outlined below.

[3] Eclipse Homepage, http://www.eclipse.org

2.2.1 Development Capabilities

2.2.1.1 Fuselet Scripting Language

The Fuselet Scripting Language (FSL) is a simple scripting language that provides an easy and convenient way to write simple programs and access the CAPI. The FSL is based on the dynamic, object-oriented programming language Python. Python is a scripting language that provides a number of important features. It has a simple syntax and programming model for novice users. At the same time, it offers an extensive set of tools for experienced programmers.

2.2.1.2 Graphical Development Editor

The Graphical Development Editor (GDE) is a multi-page editor that allows a user to develop fuselets and fuselet components graphically (see Figure 2). The user creates fuselets by "dragging and dropping" and connecting icons representing fuselet and programming constructs in a graphical editor or by editing the FSL code directly in a text editor. The GDE addresses the need to support a range of users that need to create new fuselets and have varying experience in programming/scripting.

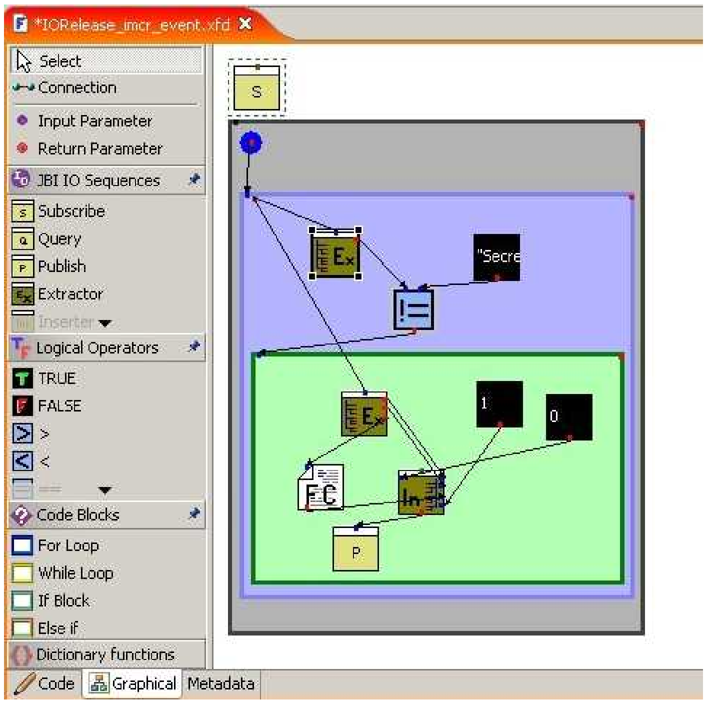


Figure 2: Graphical Development Editor

2.2.1.3 Editing Views

FDE "views" are child windows within the workbench that provides information on some resource and/or data object and may allow for the editing/changing the value of the selected resource. These views are used in conjunction with the GDE to allow the user to select a component and edit the value of that particular component. These views give the user another graphical interface to the underlying FSL without actually editing the fuselet script. Some of these views are shown in Figure 1. These views are the Port/Value Builder, Predicate Builder, and Project Navigator (surrounding the GDE).

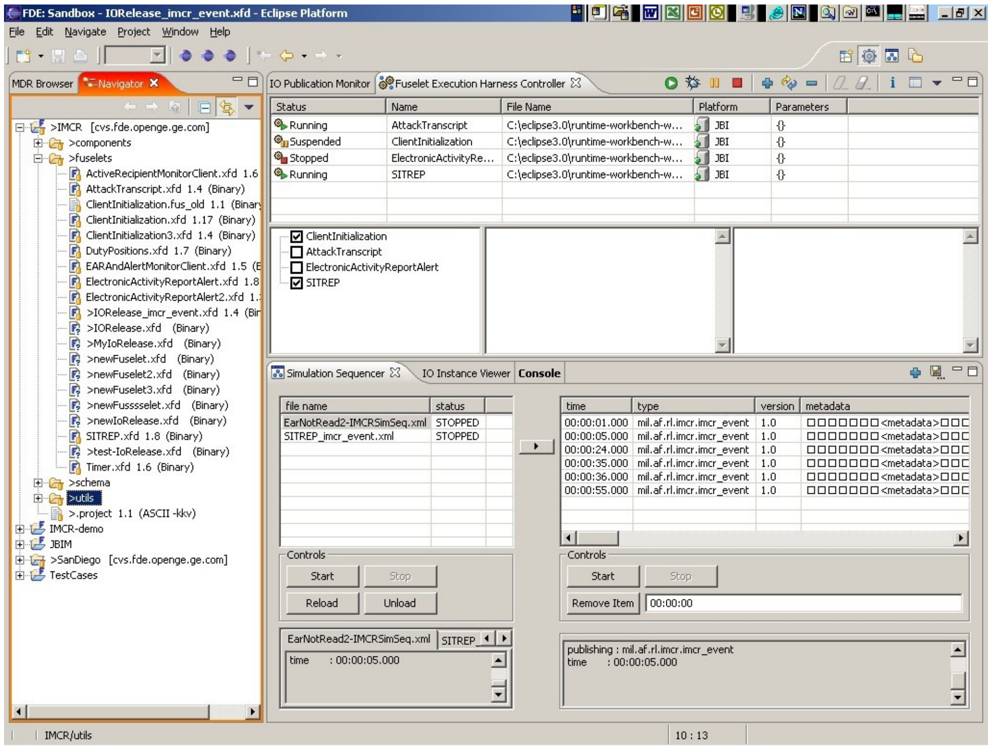
2.2.1.4 Wizards

A “wizard” is a utility that provides a sequence of dialogs which step the user through a set of tasks required to accomplish some activity. The FDE has a number of wizards that aids the user in the creation, development, and testing of fuselets. The FDE wizards are:

1. New Fuselet – creation of a new fuselet
2. New Fuselet Component – creation of a new fuselet component. These are the helper/generic components used within fuselets
3. New Simulation Sequence file - new template for the Simulation Sequencer utility.
4. New IO Schema file – new XML schema template for a new Information Object type.
5. New Fuselet Project – new fuselet project and basic directory structure described above.

2.2.2 Runtime Capabilities

The second mode of the FDE is the runtime mode used for exercising and debugging the fuselets created during development. Below is a description of the three major utilities used in runtime mode.



```
>IMCR  [cvs.fde.openge.ge.com]
  >components
  >fuselets
    ActiveRecipientMonitorClient.xfd 1.6
    AttackTranscript.xfd 1.4 (Binary)
    ClientInitialization.fus_old 1.1 (Binary
    ClientInitialization.xfd 1.17 (Binary)
    ClientInitialization3.xfd 1.4 (Binary)
    DutyPositions.xfd 1.7 (Binary)
    EARAndAlertMonitorClient.xfd 1.5 (E
    ElectronicActivityReportAlert.xfd 1.8
    ElectronicActivityReportAlert2.xfd 1.
    >IORelease_imcr_event.xfd 1.4 (Bir
    >IORelease.xfd   (Binary)
    >MyIoRelease.xfd   (Binary)
    >newFuselet.xfd   (Binary)
    >newFuselet2.xfd   (Binary)
    >newFuselet3.xfd   (Binary)
    >newFusssselet.xfd   (Binary)
    >newIoRelease.xfd   (Binary)
    SITREP.xfd 1.8 (Binary)
    >test-IoRelease.xfd   (Binary)
    Timer.xfd 1.6 (Binary)
  >schema
  >utils
  >.project 1.1 (ASCII -kkv)
IMCR-demo
JBIM
>SanDiego  [cvs.fde.openge.ge.com]
TestCases
```

Status	Name	File Name	Platform	Parameters
Running	AttackTranscript	C:\eclipse3.0\runtime-workbench-w...	JBI	{}
Suspended	ClientInitialization	C:\eclipse3.0\runtime-workbench-w...	JBI	{}
Stopped	ElectronicActivityRe...	C:\eclipse3.0\runtime-workbench-w...	JBI	{}
Running	SITREP	C:\eclipse3.0\runtime-workbench-w...	JBI	{}

- [✓] ClientInitialization
- [ ] AttackTranscript
- [ ] ElectronicActivityReportAlert
- [✓] SITREP

file name	status
EarNotRead2-IMCRSimSeq.xml	STOPPED
SITREP_imcr_event.xml	STOPPED

time	type	version	metadata
00:00:01.000	mil.af.rl.imcr.imcr_event	1.0	□□□□□□□<metadata>□□□
00:00:05.000	mil.af.rl.imcr.imcr_event	1.0	□□□□□□□<metadata>□□□
00:00:24.000	mil.af.rl.imcr.imcr_event	1.0	□□□□□□□<metadata>□□□
00:00:35.000	mil.af.rl.imcr.imcr_event	1.0	□□□□□□□<metadata>□□□
00:00:36.000	mil.af.rl.imcr.imcr_event	1.0	□□□□□□□<metadata>□□□
00:00:55.000	mil.af.rl.imcr.imcr_event	1.0	□□□□□□□<metadata>□□□

00:00:00

EarNotRead2-IMCRSimSeq.xml | SITREP_

```
time      : 00:00:05.000
```

```
publishing : mil.af.rl.imcr.imcr_event
time       : 00:00:05.000
```

Figure 3: FDE Sandbox (runtime and testing)

2.2.2.1 Fuselet Execution Harness

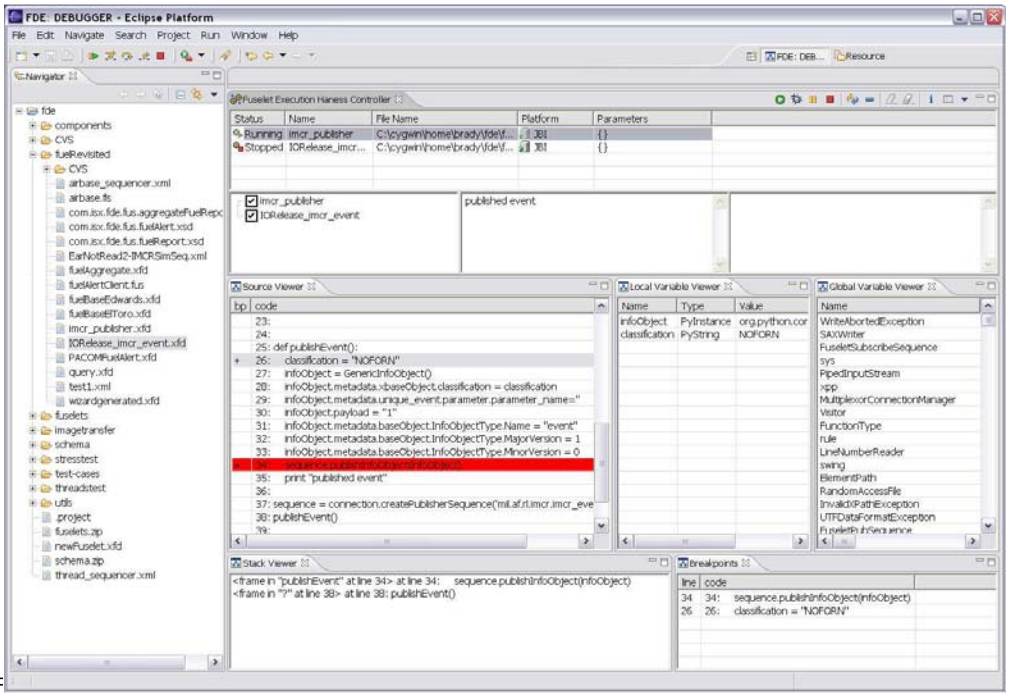
The Fuselet Execution Harness is a FDE view that contains the underlying infrastructure that allows a user to execute and test fuselets. The Fuselet Execution Harness can execute fuselets in two modes. One mode will allow all publish, subscribe, and query through a JBI. The other mode publishes, subscribes, and queries through a locally run object cache. The Object Cache is essentially a mini JBI that has most of the basic functionality of a production JBI but is primarily used in situations where a JBI has not been set up. The Execution Harness allows the user to manage currently loaded fuselets.

2.2.2.2 Simulation Sequencer

The Simulation Sequencer is a tool for managing and executing scripts that publish information objects to a JBI. The Simulation Sequencer scripts consist of publication events. Events contain a timestamp and information object instance data. The timestamp determines when the event will publish its information object to the JBI. The Simulation Sequencer contains controls, which allow easy “playback” of the scripts. The Simulation Sequencer is used to publish information objects to exercise the fuselets that are running in the Execution Harness.

2.2.2.3 Fuselet Debugger

The fuselet debugger is a full-featured debugger for the execution harness. It contains a series of views that display information about your currently running fuselet. The debugger is a very useful tool that aids in identifying problems that may not be apparent when writing a fuselet. The debugger also supports the ability to debug multiple fuselets from the execution harness at the same time.

=


Status	Name	File Name	Platform	Parameters
Running	imcr_publisher	C:\cygwin\home\brady\fde\f...	JBI	{}
Stopped	IORelease_imcr...	C:\cygwin\home\brady\fde\f...	JBI	{}

- [✓] imcr_publisher
- [✓] IORelease_imcr_event

```
published event
```

```
bp  code
    23:
    24:
    25: def publishEvent():
*   26:     classification = "NOFORN"
    27:     infoObject = GenericInfoObject()
    28:     infoObject.metadata.xbaseObject.classification = classification
    29:     infoObject.metadata.unique_event.parameter.parameter_name="
    30:     infoObject.payload = "1"
    31:     infoObject.metadata.baseObject.InfoObjectType.Name = "event"
    32:     infoObject.metadata.baseObject.InfoObjectType.MajorVersion = 1
    33:     infoObject.metadata.baseObject.InfoObjectType.MinorVersion = 0
*   34:     sequence.publishInfoObject(infoObject)
    35:     print "published event"
    36:
    37: sequence = connection.createPublisherSequence('mil.af.rl.imcr.imcr_eve
    38: publishEvent()
    39:
```

Name	Type	Value
infoObject	PyInstance	org.python.cor
classification	PyString	NOFORN

Name
WriteAbortedException
SAXWriter
FuseletSubscribeSequence
sys
PipedInputStream
xpp
MultiplexorConnectionManager
Visitor
FunctionType
rule
LineNumberReader
swing
ElementPath
RandomAccessFile
InvalidXPathException
UTFDataFormatException
FuseletPubSequence

```
<frame in "publishEvent" at line 34> at line 34:     sequence.publishInfoObject(infoObject)
<frame in "?" at line 38> at line 38: publishEvent()
```

line	code
34	34: sequence.publishInfoObject(infoObject)
26	26: classification = "NOFORN"

Figure 4: FDE Fuselet Debugger

2.3 FDE Project Accomplishments

2.3.1 Graphical fuselet development and testing

The goal of the FDE was to create a rich, user-friendly graphical environment where domain experts, with limited programming skill, can easily create, test, and deploy fuselets for the JBI. The FDE employs a graphical "drag and drop" paradigm that enables users to manipulate fuselets, whether it's in the editing of the underlying script or the ability to execute and monitor the fuselet. The FDE delivers a rich user experience in an all-in-one environment for creating and managing fuselets.

2.3.2 Fuselets

Throughout the FDE project, the GE/ISX team worked closely with the members of the AFRL JBI Fuselet research team in defining fuselet capabilities and functionality. Along with working with the AFRL JBI Fuselet research team, the GE/ISX team created a number of functional fuselets and components for FDE project demonstrations and AFRL JBI program demonstrations.

- **Demo scenario fuselets –** to test and demonstrate the capabilities of the FDE throughout the research and development of the environment, the GE/ISX team created a principle demo scenario, called "San Diego." For the San Diego scenario, many fuselet, fuselet components, and other FSL based utilities were created to help test the FDE and was delivered as an example scenario for user experimentation.

- **Information Management for Crisis Response (IMCR)** - The GE/ISX team crated 20+ fuselets and components. The fuselets created helped to demonstrate the adaptation and integration of an operational alerting and messaging system, Digital Dashboard, into the AFRL JBI platform and the Infosphere paradigm. The fuselets helped to monitor, aggregate, summarize, alert, and process information objects that the Digital Dashboard system required. The fuselets added functionality to the Digital Dashboard system without having the make a great deal of modifications to the Digital Dashboard's underlying source code. The IMCR scenario demonstrated the FDE's fuselet creation, editing, test, and debug capabilities in an operational situation.

2.3.3 Project meetings and demonstrations

Throughout the FDE project, the GE/ISX team attended meeting at AFRL Rome Research Site (RSS) and hosted meetings at the GE Global Research Center (Niskayuna, NY) to present and report on the FDE project progress, FDE demonstrations, and FDE training sessions for members of the AFRL JBI program. The GE/ISX team also supported the AFRL JBI Fuselet research program by attending and participating in AFRL JBI Principal Investigator (PI)meetings to report on and demonstrate the capabilities of the FDE. Meetings of significance that the GE/ISX team participated were:

- AFRL Science Advisory Board Poster Session (2003)
- AFRL JBI PI meetings (2004, 2005)
- AFRL JBI Minnowbrook Meeting (2004)

3 Lessons Learned and Suggestions

The following is a list of the most important issues and challenges discovered while researching and developing the FDE and a brief description of the lessons learned and suggestions for future work and activities.

3.1 JBI Platform (Core Services) 1.2.5.0 release schedule

Issue: The delayed release schedule for the JBI Platform Core Services (1.2.5.0). These delays in the delivery of the platform impacted the development and delivery schedule of the FDE to a great degree. The feature changes to the JBI platform in 1.2.5.0 included changes to the CAPI (version 1.0 to 1.5), changes to the Information Object (IO) base object model, and IO functionality within the JBI. These were substantial functional changes and required equally substantial changes in FDE core functionality.

Lesson Learned: The GE/ISX team was kept abreast of the core changes to the JBI technology, but without having the actual libraries and new JBI platform, changes could not start within the FDE. One approach would have been to work much more closely with the AFRL JBI core services development team, but this would have impacted the FDE project budget. The key lesson learned was to be much more flexible in planning and budgeting.

3.2 Jython libraries for runtime and script parsing (editors)

Issue: The version of Jython (java implementation of the Python runtime and interpreter) that is delivered as apart of the FDE 1.2.5.0 is Jython 2.1. Jython 2.1 is limited in its functionality in its pre-compiled binary distribution. This distribution has limitation on is packaged parser and is not delivered with Abstract Syntax Tree (AST) objects, which is needed by the FDE's Graphical Development Editor. For this reason, the Jython source code was needed to create the AST objects (generated via the Java Compiler Compiler, Java CC, tool). Once created, there were some simple modifications necessary to integrate the parser functionality into an Eclipse structured text editor. This generated and modified code is now "specialized" and packaged as apart of the FDE FSL editor. The newest version of Jython 2.2, is packaged with the optional classes precompiled, but there are enough changes in the underlying runtime that would require a significant refactoring effort in the FDE source code. Also, Jython 2.2 is in "alpha" status and is not stable and should not be considered while it is still in "alpha" status.

Lesson Learned: Though the Jython 2.1 runtime environment is a stable environment for the Fuselet runtime, it didn't lend itself as an easy to use library to build an eclipse editor from with its default, precompiled parser. As soon as Jython 2.2 is available as a stable release, an effort should be taken to refactor the FDE to upgrade to Jython 2.2, eliminating the need for "specialized" code for the editor.

3.3 XML/XSD parsers for IO Schema

Issue: The JBI platform Information Objects (IO) are defined using eXtensible Markup Language (XML) Schema (XSD). These schemas are available for inspection and viewing through a JBI connection using the CAPI. It is difficult for a user to read and understand even the simplest XML Schema as plain text. Using XML Schema editors, such as the Eclipse Web Tools Platform Schema Editor or XML Spy, makes it easier for users to understand the schema, but still difficult to understand how an instance of the schema, XML data, will look. The tools provided in the FDE (IO Schema Viewer, Port/Value Builder, and Predicate Builder) take an IO schema and attempts to create an XML instance object (tree) structure that adheres to the IO schema. This representation is friendlier to the FDE user, especially a user that is not familiar with raw XML text and XML schema. The present FDE capability relies on a simple algorithm to produce the XML instance data based on an IO schema. This functionality is acceptable for sufficiently simple IO schema. For more complicated IO schema the present functionality is not sufficient.

Lesson Learned: The needed functionality of the FDE to produce robust XML instance data from an XML schema is a complicated task. Only a simplified version of the instance generator is developed for the current FDE. However, further redevelopment is needed when:

- More XSD features are supported to describe complex XML instances, e.g., annotations and attributes. The parsing process then becomes more complicated.
- The XSD uses features (such as group and choices) that could generate XML instances with different structures. This means a generated XML instance may

contain all possible features, therefore not adhering to the schema, but provides a richer, dynamic user interface (all possible choices).

- The type definition becomes more complicated. For example, the same type definition is used in multiple element definitions, which may cause a recursive algorithm. Complicated type definitions need a smarter parser.

The complexity of the XSD specification makes the issue much more complicated than it seems at the surface. Determining how each XSD feature should be interpreted in making an instance XML document has special consideration.

The present FDE could be further enhanced with the suggested robust schema to instance XML (XSD to XML) functionality. This robust functionality could be used in other JBI Platform tools to help render/view IO Schema and allow non-programmers and XML-savvy users to understand IO structures.

Proposed Future Direction

The following are suggestions from the GE FDE project team concerning the future directions of the FDE, both technically and programmatically.

3.4 Leverage Newer Eclipse Technologies

Over the past twenty-eight months of the FDE project, the Eclipse project technologies that are the core of the Fuselet Development Environment have become much more robust and commercial quality. Many technical articles, books, and websites have been created since the beginning of the FDE project (FDE started on Eclipse 2.0, now its built on 3.1.1) and they provide developers with excellent resources for plug-in design and development. The Eclipse Project has also introduced a number of new sub-projects and technologies that were not available during most of the FDE design and development. Some of the sub-projects and technologies could be leveraged to improve the FDE, especially in the areas of the Graphical Development Editor (GDE) and Information Object viewing tools.

3.4.1 Eclipse Modeling Framework

The Eclipse Modeling Framework (EMF) is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XML Metadata Interchange (XMI) or XSD, EMF provides tools and runtime support to produce a set of Java classes for the model, a set of adapter classes that enable viewing and basic editing. The EMF project includes a rich XML Schema Infoset Model API that is a reference library for examining, creating, and modifying XSD. The EMF has become an important sub-project for Eclipse and is the basis for many of the new Eclipse XML, Java Enterprise Edition, and Web Services development tools. The EMF could be leveraged to create more robust tools and editors for creating and editing new IO schema, Simulation Sequencer, and Fuselet Metadata files.

3.4.2 Graphical Modeling Framework

The Graphical Modeling Framework (GMF) project provides the fundamental infrastructure and components for developing visual design and modeling surfaces in Eclipse. In essence, GMF forms a generative bridge between Eclipse Modeling Framework (EMF) and Graphical Editor Framework (GEF), whereby a diagram definition will be linked to a domain model as input to the generation of a visual editor. This framework could be leveraged and used to improve the FDE GDE, where a robust fuselet model could be defined and a more eclipse native editor could be created, mostly from generated code from the fuselet model. This technology is in its incubator stage, but shows much promise.

3.5 FDE in the Fuselet Community

The FDE design and user interface has matured over the course of the project. User input and feedback were critical in developing and modifying the GDE and supporting Infosphere Platform tools. To support evolution of the FDE and its usefulness in fuselet research and development, members of the Fuselet Community must use it and provide as much feedback as possible. Fuselet community members are the researchers and developers of current Fuselet technologies and can benefit the most by its improvements. By allowing the Fuselet Community access, AFRL will obtain more feedback regarding the tools and shape the future of the FDE. This access may include the availability of the underlying source code in which the community could modify, enhance, and extend. This is similar to other open source projects, such as Linux and Eclipse, where the community of users creates a “grass roots” effort to improve the project.

4 Conclusion

In summary, this report provides a high level overview of a Fuselet Development Environment (FDE) developed by the GE/ISX team, and suggests that further research and development could be pursued to improve certain aspects of functionality and incorporate additional features in large part due to the rapid evolution of the technology supporting the FDE. This includes evolution of the Eclipse framework, Jython, and AFRL’s JBI reference implementation. Regardless of the potential for continuing enhancements, the FDE project has successfully demonstrated the development of a fuselet production environment geared toward fuselet authors ranging from novice to highly skilled programmers.